AMENDMENTS TO THE CLAIMS:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing Of Claims:

Please amend the claims as follows:

1. (Currently Amended) A method for generating an integer part of a logarithm of a floating point operand comprising:

analyzing, using a floating point unit, a format of the floating point operand and generating at least one signal representative of the format;

determining a magnitude of an unbiased exponent of the floating point operand as an intermediate result based on the at least one signal, wherein the unbiased exponent is represented by unbiased exponent bits;

determining an exponent field of the intermediate result ;

determining a fraction field high part of the intermediate result; and

assembling a result equal to the integer part of the logarithm of the floating point operand based on the at least one signal wherein, if the floating point operand is in at least one of a denormalized format, a normalized non-zero format, and a delimited format, an exponent field of the result equals the exponent field of the intermediate result and a fraction field high part of the result equals the fraction field high part of the intermediate result.

2. (Original) The method of claim 1, wherein determining the magnitude of the unbiased exponent of the floating point operand as the intermediate result comprises:

determining a magnitude of a difference between exponent field bits of the floating point operand and a predetermined bias, if the format is a normalized non-zero format; and

identifying a number of leading zeros in fraction field bits of the floating point operand and adding the number to the predetermined bias, if the format is a denormalized format.

3. (Original) The method of claim 1, wherein determining the magnitude of the unbiased exponent of the floating point operand as the intermediate result comprises:

identifying a number of trailing zeros in fraction field bits of the floating point operand and adding the number to a predetermined bias, if the format is a delimited format.

4. (Original) The method of claim 1, wherein determining the exponent field of the intermediate result comprises:

identifying a number of leading zeros in the unbiased exponent of the floating point operand and subtracting the number from a predetermined bit pattern to determine the exponent field of the intermediate result.

5. (Original) The method of claim 1, wherein determining the fraction field high part of the intermediate result comprises:

normalizing bits of the unbiased exponent of the floating point operand by determining a number of leading zeros in the unbiased exponent bits, shifting the unbiased exponent bits to the left by the number of leading zeros, and

discarding a most significant bit from the normalized unbiased exponent bits.

6. (Original) The method of claim 1, wherein assembling the result comprises:

setting a sign bit of the result based on the at least one signal representative of the format of the operand;

setting exponent field bits of the result based on the at least one signal representative of the format of the floating point operand; and

setting fraction field bits of the result based on the at least one signal representative of the format of the operand.

7. (Original) The method according to claim 6, wherein setting the sign bit comprises:

setting the sign bit to a first value indicating a negative result if the format is at least one of a negative NaN format, a zero format, an underflow format, and a denormalized format; and

setting the sign bit to a second value indicating a positive result otherwise.

8. (Original) The method according to claim 6, wherein setting the exponent field bits of the result comprises:

setting the exponent field bits of the result to zero if the format of the floating point operand is a normalized non-zero format and the magnitude of the unbiased exponent of the floating point operand is zero;

setting the exponent field bits of the result to a first bit pattern if the format of the floating point operand is at least one of an underflow format and a overflow format; and

setting the exponent field bits of the result to a second bit pattern if the format of the floating point operand is at least one of a zero format, an infinity format, and a NaN format.

9. (Original) The method according to claim 8, wherein:

the first bit pattern is representative of an overflow format.

10. (Original) The method according to claim 8, wherein:

the second bit pattern is representative of one of an infinity and a NaN format.

11. (Original) The method according to claim 6, wherein setting the fraction field bits of the result comprises:

setting the fraction field bits of the result to zero if the format of the floating point operand is a normalized non-zero format and the magnitude of the unbiased exponent of the floating point operand is zero;

setting the fraction field bits of the result to a first bit pattern if the format of the floating point operand is at least one of an underflow format and a overflow format; and

setting the fraction field bits of the result to fraction field bits of the floating point operand if the format of the floating point operand is at least one of an infinity format and a NaN format; and

setting the fraction field bits of the result to zero with the exception that at least one bit of the fraction field bits is set to a first value to indicate a divide-by-zero status, if the format of the floating point operand is a zero format.

12. (Original) The method according to claim 11, wherein:

the first bit pattern is representative of an overflow format.

13. (Original) The method according to claim 6, wherein setting the fraction field bits of the result comprises:

setting the fraction field bits of the result to zero if the format of the floating point operand is a normalized non-zero format and the magnitude of the unbiased exponent of the floating point operand equals zero;

setting the fraction field bits of the result to the fraction field bits of the floating point operand if the format of the floating point operand is at least one of an infinity format and a NaN format; and

setting the fraction field bits of the result to zero and providing at least one status signal having a first value indicating a divide-by-zero status if the format of the floating point operand is a zero format.

14. (Original) The method according to claim 1, further comprising:

embedding status information in the result.

15. (Original) The method according to claim 14, wherein embedding the status information in the result comprises:

setting at least one bit in the result to a first value to indicate a divide-by-zero status if the format of the floating point operand is a zero format.

16. (Original) The method according to claim 14, wherein embedding the status information in the result comprises:

setting at least one bit of the result that stores the status information equivalent to a corresponding bit in the floating point operand that represents the status information.

17. (Original) A computer-readable medium on which is stored a set of instructions for generating an integer part of a logarithm of a floating point operand, which when executed performs steps comprising:

analyzing a format of the floating point operand and generating at least one signal representative of the format;

determining a magnitude of an unbiased exponent of the floating point operand as an intermediate result based on the at least one signal, wherein the unbiased exponent is represented by unbiased exponent bits;

determining an exponent field of the intermediate result ;

determining a fraction field high part of the intermediate result; and

assembling a result equal to the integer part of the logarithm of the floating point operand based on the at least one signal wherein, if the floating point operand is in at least one of a denormalized format, a normalized non-zero format, and a delimited format, an exponent field of the result equals the exponent field of the intermediate result and a fraction field high part of the result equals the fraction field high part of the intermediate result.

18. (Original) The computer-readable medium of claim 17, wherein determining the magnitude of the unbiased exponent of the floating point operand as the intermediate result comprises:

determining a magnitude of a difference between exponent field bits of the floating point operand and a predetermined bias, if the format is a normalized non-zero format; and

identifying a number of leading zeros in fraction field bits of the floating point operand and adding the number to the predetermined bias, if the format is a denormalized format.

19. (Original) The computer-readable medium of claim 17, wherein determining the magnitude of the unbiased exponent of the floating point operand as the intermediate result comprises:

identifying a number of trailing zeros in fraction field bits of the floating point operand and adding the number to a predetermined bias, if the format is a delimited format.

20. (Original) The computer-readable medium of claim 17, wherein determining the exponent field of the intermediate result comprises:

identifying a number of leading zeros in the unbiased exponent of the floating point operand and subtracting the number from a predetermined bit pattern to determine the exponent field of the intermediate result.

21. (Original) The computer-readable medium of claim 17, wherein determining the fraction field high part of the intermediate result comprises:

normalizing bits of the unbiased exponent of the floating point operand by determining a number of leading zeros in the unbiased exponent bits, shifting the unbiased exponent bits to the left by the number of leading zeros, and

discarding a most significant bit from the normalized unbiased exponent bits.

22. (Original) The computer-readable medium of claim 17, wherein assembling the result comprises:

setting a sign bit of the result based on the at least one signal representative of the format of the operand;

setting exponent field bits of the result based on the at least one signal representative of the format of the floating point operand; and

setting fraction field bits of the result based on the at least one signal representative of the format of the operand.

23. (Original) The computer-readable medium of claim 22, wherein setting the sign bit comprises:

setting the sign bit to a first value indicating a negative result if the format is at least one of a negative NaN format, a zero format, an underflow format, and a denormalized format; and

setting the sign bit to a second value indicating a positive result otherwise.

24. (Original) The computer-readable medium of claim 22, wherein setting the exponent field bits of the result comprises:

setting the exponent field bits of the result to zero if the format of the floating point operand is a normalized non-zero format and the magnitude of the unbiased exponent of the floating point operand is zero;

setting the exponent field bits of the result to a first bit pattern if the format of the floating point operand is at least one of an underflow format and a overflow format; and

setting the exponent field bits of the result to a second bit pattern if the format of the floating point operand is at least one of a zero format, an infinity format, and a NaN format.

25. (Original) The computer-readable medium of claim 24, wherein:

the first bit pattern is representative of an overflow format.

26. (Original) The computer-readable medium of claim 24, wherein:

the second bit pattern is representative of one of an infinity and a NaN format.

27. (Original) The computer-readable medium of claim 22, wherein setting the fraction field bits of the result comprises:

setting the fraction field bits of the result to zero if the format of the floating point operand is a normalized non-zero format and the magnitude of the unbiased exponent of the floating point operand is zero;

setting the fraction field bits of the result to a first bit pattern if the format of the floating point operand is at least one of an underflow format and a overflow format; and

setting the fraction field bits of the result to fraction field bits of the floating point operand if the format of the floating point operand is at least one of an infinity format and a NaN format; and

setting the fraction field bits of the result to zero with the exception that at least one bit of the fraction field bits is set to a first value to indicate a divide-by-zero status, if the format of the floating point operand is a zero format.

28. (Original) The computer-readable medium of claim 27, wherein:

the first bit pattern is representative of an overflow format.

29. (Original) The computer-readable medium of claim 22, wherein setting the fraction field bits of the result comprises:

setting the fraction field bits of the result to zero if the format of the floating point operand is a normalized non-zero format and the magnitude of the unbiased exponent of the floating point operand equals zero;

setting the fraction field bits of the result to the fraction field bits of the floating point operand if the format of the floating point operand is at least one of an infinity format and a NaN format; and

setting the fraction field bits of the result to zero and providing at least one status signal having a first value indicating a divide-by-zero status if the format of the floating point operand is a zero format.

30. (Original) The computer-readable medium of claim 17, wherein the set of instructions when executed further embeds status information in the result.

31. (Original) The computer-readable medium of claim 30, wherein embedding the status information in the result comprises:

setting at least one bit in the result to a first value to indicate a divide-by-zero status if the format of the floating point operand is a zero format.

32. (Original) The computer-readable medium of claim 30, wherein embedding the status information in the result comprises:

setting at least one bit of the result that stores the status information equivalent to a corresponding bit in the floating point operand that represents the status information.

33. (Original) A system for generating an integer part of a logarithm of a floating point operand comprising:

an operand analysis unit for analyzing a format of the floating point operand and generating at least one signal representative of the format;

a processing unit coupled to the operand analysis unit, the processing unit being operative to determine a magnitude of an unbiased exponent of the floating point operand as an intermediate result based on the at least one signal, determine an exponent field of the intermediate result, and determine a fraction field high part of the intermediate result; and

a result generator coupled to the operand analysis unit and the processing unit, the result generator being operative to assemble a result equal to the integer part of the logarithm of the floating point operand based on the at least one signal, wherein, if the floating point operand is in at least one of a denormalized format, a normalized non-zero format, and a delimited format, an exponent field of the result equals the exponent field of the intermediate result and a fraction field high part of the result equals the fraction field high part of the intermediate result.

34. (Original) The system according to claim 33, wherein the floating point operand is in at least one of an underflow format and an overflow format.

35. (Original) The system according to claim 33, wherein status information is stored in the floating point operand and preserved by the result generator within the result.

36. (Original) The system according to claim 33, wherein the result generator is further operative to generate status information that is stored in the resulting operand.